Intercept and modify video frames

Description

Frame interception works only for stream publishers and gets called before frame is distributed to encoding/rescaling groups. It can be used with WebRTC and WebSocket player clients.

To intercept frames, interface `IDecodedFrameInterceptor` from package `com.flashphoner.sdk.media` should be implemented. The interface declares one method - `frameDecoded()` - allowing to process intercepted frame depending on the stream name:

```
void frameDecoded(String streamName, YUVFrame frame);
```

Recommendations

Because of the nature of real-time video stream, `frameDecoded()` method is synchronous, i.e. frame 2 cannot be decoded while you are working on frame 1. Therefore, try to avoid long computations in `frameDecoded()` to minimize impact on stream delay and smoothness.

Pixel manipulation

All pixel data is in `DirectByteBuffer` allocated in native C. It is not recommended to clone backing buffer or YUVFrame because that will lead to huge memory leak and server crash as a result. Pixel manipulation should be performed using `buffer.get()` and `buffer.put()` methods, or helper methods, such as `readPixel()` and `writePixel()`.

Configuration

1. Create a Java class for frames interception
a. Create class implementing `com.flashphoner.sdk.media.IDecodedFrameInterceptor` (see the example below)
b. Copy it to `/usr/local/FlashphonerWebCallServer/lib`
c. Run the following commands

```
- javac -classpath .:tbs-flashphoner.jar:slf4j-api-1.6.4.jar -d .
MyDecodedFrameInterceptor.java
- jar cf frame-interceptor.jar com
- rm -rf com
```

2. Add your interceptor to `flashphoner.properties`

```
decoded_frame_interceptor =com.example.video.MyDecodedFrameInterceptor
```

Usage

Publish a stream and play it as WebRTC or WebSocket, e.g. using 'Streaming Min' or 'WS Player Min' demo clients in Chrome or Firefox.

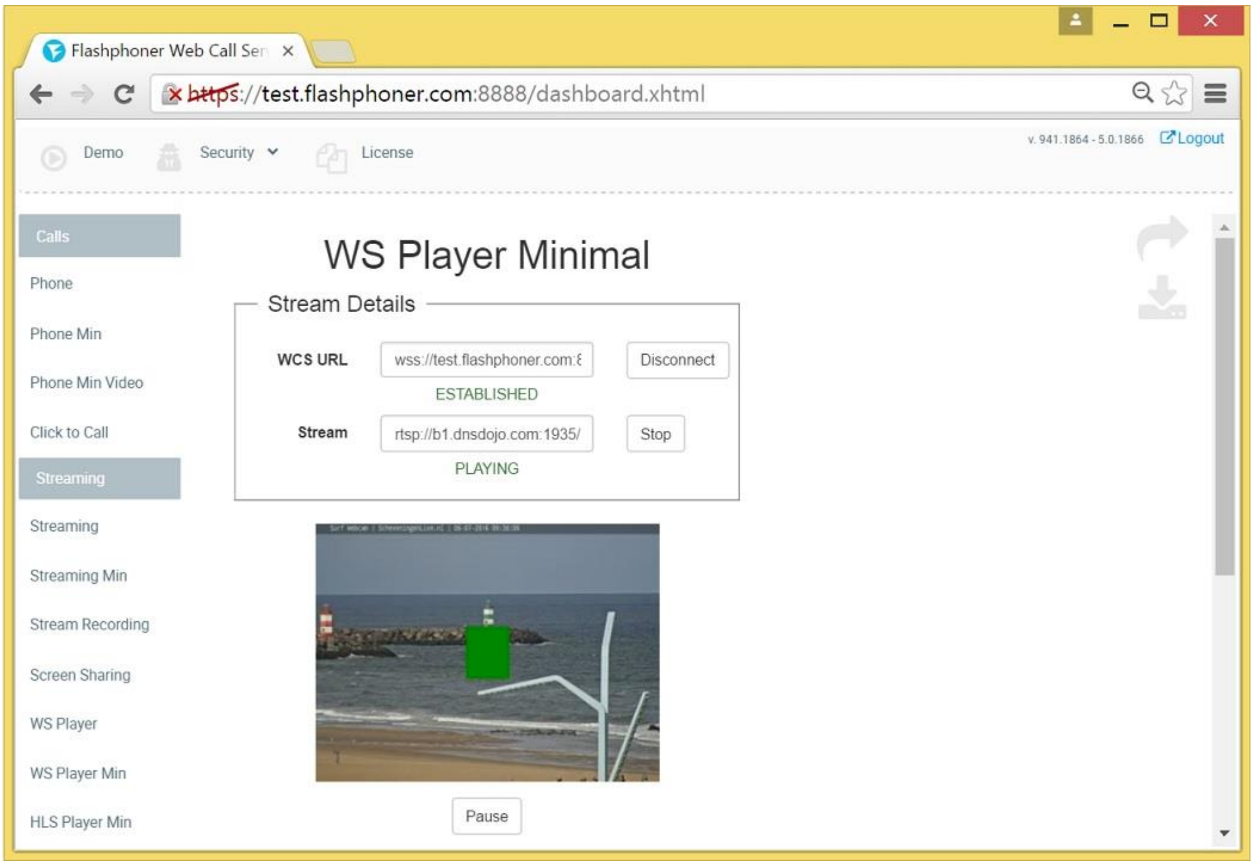
Example

Below is an example of class implementing `com.flashphoner.sdk.media.IDecodedFrameInterceptor`. It draws green square rectangle in the center of a frame.

```
package com.example.video;

import com.flashphoner.sdk.media.*;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class MyDecodedFrameInterceptor implements IDecodedFrameInterceptor {

    //create one global logger
    private static final Logger log =
LoggerFactory.getLogger("MyDecodedFrameInterceptor");

    public void frameDecoded(String streamName, YUVFrame frame) {
       log.info("Got frame " + frame);
       //draw 100x100 rectangle in the center
```

```
      int rectSide = 100;
      byte[] greenPixel = new byte[]{0x00, 0x00, 0x00};
      if (frame.getWidth() > rectSide && frame.getHeight() > rectSide) {
          int x = frame.getWidth()/2 - rectSide/2;
          int y = frame.getHeight()/2 - rectSide/2;
          int xLimit = x + rectSide;
          int yLimit = y + rectSide;
          log.info("Draw rect x:" + x + "-" + xLimit + " y:" + y + "-" + yLimit);
          for (; x < xLimit; x++) {
             for (int y2 = y; y2 < yLimit; y2++) {
                 frame.writePixel(x, y2, greenPixel);
             }
          }
      }
    }
}
```

On the screenshot below RTSP stream is played as WebSocket in Chrome when decoded frame interceptor is set to the `MyDecodedFrameInterceptor`.



WS Player Minimal

Stream Details

WCS URL wss://test.flashphoner.com:8
ESTABLISHED

Stream rtsp://b1.dnsdojo.com:1935/
PLAYING